CS425 – DATABASE ORGANIZATION

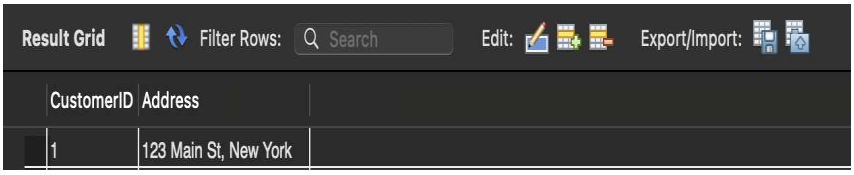
PROJECT DELIVERABLE 3 – Test a variety of SQL queries.

Team Members

Name	CWID
Vivek Saka	A20554037
Bhavana Pandey	A20554338
Akhil Kumar Marni	A20554334
Akash Thirumuruganantham	A20539883

1. **Basic Select Query: Retrieve the details of a specificcustomerID**

```
SELECT CustomerID, Address
FROM Customers
WHERE LastName = 'Doe' AND FirstName = 'John';
```

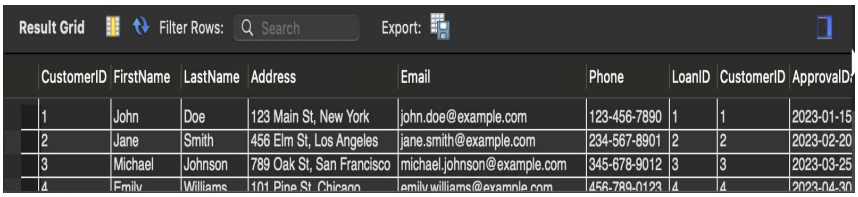


Result Grid Filter Rows: Search Edit: Export/Import:

CustomerID	Address
1	123 Main St, New York

2. **Joining Tables : Join Customers and Loan table based on CustomerID**

```
SELECT *
FROM Customers
INNER JOIN Loan ON Customers.CustomerID = Loan.CustomerID;
```



Result Grid Filter Rows: Search Export:

CustomerID	FirstName	LastName	Address	Email	Phone	LoanID	CustomerID	ApprovalD
1	John	Doe	123 Main St, New York	john.doe@example.com	123-456-7890	1	1	2023-01-15
2	Jane	Smith	456 Elm St, Los Angeles	jane.smith@example.com	234-567-8901	2	2	2023-02-20
3	Michael	Johnson	789 Oak St, San Francisco	michael.johnson@example.com	345-678-9012	3	3	2023-03-25
4	Emily	Williams	101 Pine St, Chicago	emily.williams@example.com	456-789-0123	4	4	2023-04-30

3. **Aggregation Query: Find the total number loans taken by each customers.**

```
SELECT Customers.CustomerID,
Customers.FirstName,
Customers.LastName,
      COUNT(Loan.LoanID) AS TotalLoans
FROM Customers
LEFT JOIN Loan ON Customers.CustomerID = Loan.CustomerID
GROUP BY Customers.CustomerID, Customers.FirstName, Customers.LastName;
```

Result Grid Filter Rows: Search Export:

CustomerID	FirstName	LastName	TotalLoans
1	John	Doe	2
2	Jane	Smith	2
3	Michael	Johnson	2
4	Emily	Williams	2
5	Christopher	Brown	2
6	Jessica	Jones	2
7	David	Garcia	2
8	Sarah	Martinez	2
9	Andrew	Hernandez	2

4. **Subquery with Aggregation: Retrieve the Customer with the highest Loan.**

```
SELECT Customers.CustomerID,
Customers.FirstName,
Customers.LastName,
Loan.Amount
FROM Customers
INNER JOIN Loan ON Customers.CustomerID = Loan.CustomerID
WHERE Customers.CustomerID = (
   SELECT CustomerID
   FROM Loan
   GROUP BY CustomerID
   ORDER BY SUM(Amount) DESC
   LIMIT 1
);
```

Result Grid Filter Rows: Search Export:

CustomerID	FirstName	LastName	Amount
25	Joseph	Phillips	2600000.00

5. **Window Function - Ranking:Retrieve the Rankings of Customers with the highest Loan.**

```
SELECT CustomerID, FirstName, LastName, Amount, RANK() OVER (ORDER BY
TotalLoanAmount DESC) AS loan_rank
FROM (
   SELECT Customers.CustomerID,
Customers.FirstName,
Customers.LastName,
Loan.Amount,
         SUM(Loan.Amount) OVER (PARTITION BY Customers.CustomerID) AS
TotalLoanAmount
   FROM Customers
   INNER JOIN Loan ON Customers.CustomerID = Loan.CustomerID
) AS ranked_data;
```

Result Grid Filter Rows: Search Export:

CustomerID	FirstName	LastName	Amount	loan_rank
25	Joseph	Phillips	2600000.00	1
24	Barbara	Turner	2500000.00	2
23	Mark	Roberts	2400000.00	3
22	Karen	Perez	2300000.00	4
21	Charles	Mitchell	2200000.00	5
20	Linda	Carter	2100000.00	6
19	William	Nelson	2000000.00	7
18	Elizabeth	Gonzalez	1900000.00	8
17	Richard	Baker	1800000.00	9
16	Patricia	Adams	1700000.00	10
15	Matthew	Green	1600000.00	11
14	Jennifer	Scott	1500000.00	12

6. **Window Function - Cumulative Sum:payments made by each customer in the calculation of the cumulative sum of loan amounts.**

```
SELECT L.CustomerID,
C.FirstName,
C.LastName,
L.Amount,
      SUM(L.Amount) OVER (PARTITION BY L.CustomerID ORDER BY L.LoanID
ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS
CumulativeLoanSum,
      COALESCE(SUM(P.Amount) OVER (PARTITION BY P.CustomerID ORDER BY
P.PaymentDate ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT
ROW), 0) AS CumulativePaymentSum
FROM Loan L
INNER JOIN Customers C ON L.CustomerID = C.CustomerID
LEFT JOIN Payments P ON L.CustomerID = P.CustomerID AND L.LoanID =
P.LoanID;
```

Result Grid Filter Rows: Search Export:

CustomerID	FirstName	LastName	Amount	CumulativeLoanSum	CumulativePaymentSu...
1	John	Doe	200000.00	200000.00	5000.00
2	Jane	Smith	300000.00	300000.00	6000.00
3	Michael	Johnson	400000.00	400000.00	7000.00
4	Emily	Williams	500000.00	500000.00	8000.00
5	Christopher	Brown	600000.00	600000.00	9000.00
6	Jessica	Jones	700000.00	700000.00	10000.00
7	David	Garcia	800000.00	800000.00	11000.00
8	Sarah	Martinez	900000.00	900000.00	12000.00
9	Andrew	Hernandez	1000000.00	1000000.00	13000.00
10	Lisa	Lopez	1100000.00	1100000.00	14000.00
11	Daniel	Young	1200000.00	1200000.00	15000.00
12	Mary	King	1300000.00	1300000.00	16000.00

Result 11

7. **Subquery with EXISTS: Identify customers who have taken out loans**

```
SELECT CustomerID,
      FirstName,
LastName
FROM Customers C
WHERE EXISTS (
   SELECT 1
   FROM Loan L
   WHERE L.CustomerID = C.CustomerID
);
```

Result Grid Filter Rows: Search Edit: Export/Import:

CustomerID	FirstName	LastName
1	John	Doe
2	Jane	Smith
3	Michael	Johnson
4	Emily	Williams
5	Christopher	Brown
6	Jessica	Jones
7	David	Garcia
8	Sarah	Martinez
9	Andrew	Hernandez
10	Lisa	Lopez
11	Daniel	Young
12	Mary	King

8. **Common Table Expression (CTE): calculate the loan amount, total payments made, and the remaining balance for each customer.**

```
WITH CustomerLoanInfo AS (
```

```
    SELECT C.CustomerID,
C.FirstName,
C.LastName,
L.Amount AS LoanAmount,
        SUM(P.Amount) AS TotalPayments,
L.Amount - COALESCE(SUM(P.Amount), 0) AS RemainingBalance
    FROM Customers C
    LEFT JOIN Loan L ON C.CustomerID = L.CustomerID
    LEFT JOIN Payments P ON L.LoanID = P.LoanID
    GROUP BY C.CustomerID, C.FirstName, C.LastName, L.Amount
)
SELECT *
FROM CustomerLoanInfo;
```

Result Grid Filter Rows: Search Export:

CustomerID	FirstName	LastName	LoanAmount	TotalPayments	RemainingBalance
1	John	Doe	200000.00	5000.00	195000.00
2	Jane	Smith	300000.00	6000.00	294000.00
3	Michael	Johnson	400000.00	7000.00	393000.00
4	Emily	Williams	500000.00	8000.00	492000.00
5	Christopher	Brown	600000.00	9000.00	591000.00
6	Jessica	Jones	700000.00	10000.00	690000.00
7	David	Garcia	800000.00	11000.00	789000.00
8	Sarah	Martinez	900000.00	12000.00	888000.00
9	Andrew	Hernandez	1000000.00	13000.00	987000.00
10	Lisa	Lopez	1100000.00	14000.00	1086000.00
11	Daniel	Young	1200000.00	15000.00	1185000.00
12	Mary	King	1300000.00	16000.00	1284000.00

Result 13

9. **Window Function - Lead and Lag: Show the previous and nextpayemnts made by a customer**

```
WITH LoanPaymentDetails AS (
    SELECT
L.CustomerID,
```

```
L.LoanID,
L.Amount AS LoanAmount,
P.Amount AS PaymentAmount,
L.ApprovalDate,
P.PaymentDate,
        LAG(P.Amount) OVER (PARTITION BY L.CustomerID, L.LoanID ORDER BY
P.PaymentDate) AS PreviousPaymentAmount,
        LEAD(P.Amount) OVER (PARTITION BY L.CustomerID, L.LoanID ORDER BY
P.PaymentDate) AS NextPaymentAmount,
C.FirstName,
C.LastName
    FROM
        Loan L
    JOIN
        Payments P ON L.LoanID = P.LoanID
    JOIN
        Customers C ON L.CustomerID = C.CustomerID
)
SELECT
CustomerID,
LoanID,
    FirstName,
LastName,
LoanAmount,
PaymentAmount,
ApprovalDate,
PaymentDate,
PreviousPaymentAmount,
NextPaymentAmount
FROM
LoanPaymentDetails;
```

Result Grid Filter Rows: Search Export:

CustomerID	LoanID	FirstName	LastName	LoanAmount	PaymentAmount	ApprovalDate	PaymentDate	PreviousPaymentAmou...	NextPaymentAmou...
1	1	John	Doe	200000.00	5000.00	2023-01-15	2023-02-01	NULL	3.00
1	1	John	Doe	200000.00	3.00	2023-01-15	2024-02-20	5000.00	16.00
1	1	John	Doe	200000.00	16.00	2023-01-15	2024-02-22	3.00	6.00
1	1	John	Doe	200000.00	6.00	2023-01-15	2024-02-23	16.00	18.00
1	1	John	Doe	200000.00	18.00	2023-01-15	2024-02-29	6.00	12.00
1	1	John	Doe	200000.00	12.00	2023-01-15	2024-03-04	18.00	19.00
1	1	John	Doe	200000.00	19.00	2023-01-15	2024-03-14	12.00	NULL
2	2	Jane	Smith	300000.00	6000.00	2023-02-20	2023-03-01	NULL	4.00
2	2	Jane	Smith	300000.00	4.00	2023-02-20	2024-02-16	6000.00	8.00

10. **OLAP - Running Total:calculates the running total of the Amount column for each customer**

```
SELECT
CustomerID,
PaymentDate,
   Amount,
   SUM(Amount) OVER (PARTITION BY CustomerID ORDER BY PaymentDate) AS
RunningTotal
FROM
   Payments
ORDER BY
CustomerID, PaymentDate;
```

Result Grid Filter Rows: Search Export:

CustomerID	PaymentDate	Amount	RunningTotal
1	2023-02-01	5000.00	5000.00
1	2024-02-20	3.00	5003.00
1	2024-02-22	16.00	5019.00
1	2024-02-23	6.00	5025.00
1	2024-02-29	18.00	5043.00
1	2024-03-04	12.00	5055.00
1	2024-03-14	19.00	5074.00
2	2023-03-01	6000.00	6000.00
2	2024-02-16	4.00	6012.00

11. **OLAP - Total number of loans taken for each property type including payments made for that property**

```
SELECT
Property.Type AS PropertyType,
   COUNT(DISTINCT Loan.LoanID) AS TotalLoans,
   SUM(Payments.Amount) AS TotalPayments
FROM
   Property
LEFT JOIN
   Loan ON Property.PropertyID = Loan.PropertyID
LEFT JOIN
   Payments ON Loan.LoanID = Payments.LoanID
```

```
GROUP BY
Property.Type
ORDER BY
Property.Type;
```

Result Grid Filter Rows: Search Export:

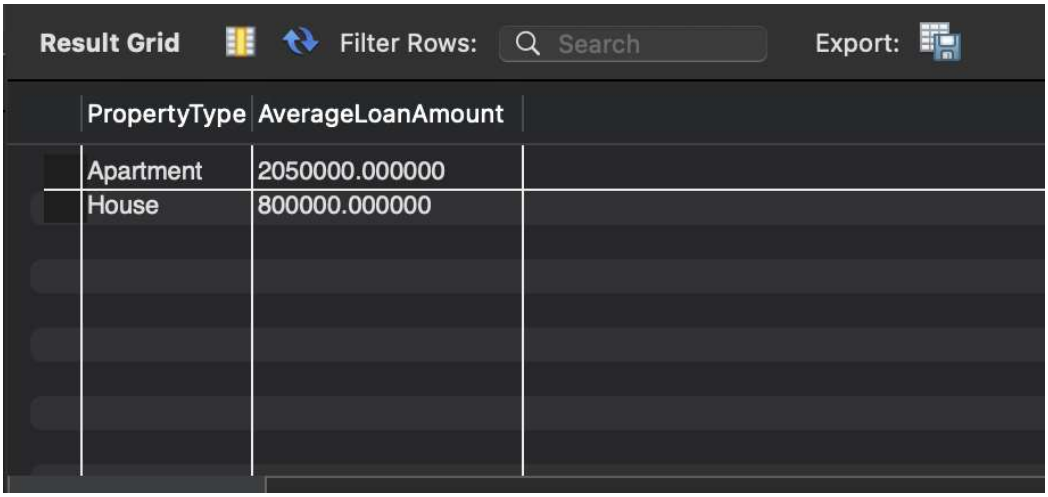
PropertyType	TotalLoans	TotalPayments
Apartment	12	282724.00
House	13	143811.00

Result 24

12. Aggregation: Find average number of loan amount taken given property type

```
SELECT
Property.Type AS PropertyType,
   AVG(Loan.Amount) AS AverageLoanAmount
FROM
   Property
LEFT JOIN
   Loan ON Property.PropertyID = Loan.PropertyID
GROUP BY
Property.Type
ORDER BY
```

```
Property.Type;
```



PropertyType	AverageLoanAmount
Apartment	2050000.000000
House	800000.000000

13. DENSE_RANK() - Rank customers with highest loan

```
SELECT
CustomerID,
   FirstName,
LastName,
DenseRankHighestLoan
FROM (
   SELECT
c.CustomerID,
c.FirstName,
c.LastName,
      DENSE_RANK() OVER (ORDER BY maxLoanAmount DESC) AS
DenseRankHighestLoan
   FROM
      Customers c
   JOIN (
      SELECT
CustomerID,
         MAX(Amount) AS maxLoanAmount
      FROM
         Loan
      GROUP BY
CustomerID
   ) l ON c.CustomerID = l.CustomerID
) AS Subquery;
```

Result Grid Filter Rows: Search Export:

CustomerID	FirstName	LastName	DenseRankHighestLoan
25	Joseph	Phillips	1
24	Barbara	Turner	2
23	Mark	Roberts	3
22	Karen	Perez	4
21	Charles	Mitchell	5
20	Linda	Carter	6
19	William	Nelson	7
18	Elizabeth	Gonzalez	8
17	Richard	Baker	9

Result 31

14. Retrieve customers who have made payments falling within a certain range

```
SELECT
CustomerID,
   COUNT(*) AS NumberOfPayments,
   SUM(Amount) AS TotalPayments
FROM
   Payments
GROUP BY
CustomerID
HAVING
   SUM(Amount) BETWEEN 10000 AND 15000;
```

Result Grid Filter Rows: Search Export:

CustomerID	NumberOfPaymen...	TotalPayments
6	7	10068.00
7	7	11061.00
8	7	12063.00
9	7	13052.00
10	7	14054.00

15. density of customers choosing a bank

```
SELECT
Bank.BankID,
Bank.Name AS BankName,
   COUNT(DISTINCT Customers.CustomerID) AS NumberOfCustomers
FROM
   Bank
LEFT JOIN
   Loan ON Bank.BankID = Loan.BankID
LEFT JOIN
   Customers ON Loan.CustomerID = Customers.CustomerID
GROUP BY
Bank.BankID, Bank.Name
ORDER BY
NumberOfCustomersDESC;
```

Result Grid Filter Rows: Search Export:

BankID	BankName	NumberOfCustom...
2	Chase Bank	5
3	Wells Fargo	5
4	Citi Bank	2
1	Bank of America	1
5	TD Bank	1
6	HSBC Bank	1
7	Bank of China	1
8	Barclays	1
9	Credit Suisse	1

16. Find which customers are close to closing the loan.

```
SELECT
C.CustomerID,
C.FirstName,
C.LastName,
L.LoanID,
   GREATEST(CEIL((L.Duration - DATEDIFF(CURDATE(), L.ApprovalDate)) / 30), 0)
AS RemainingDurationMonths
FROM
   Customers C
JOIN
   Loan L ON C.CustomerID = L.CustomerID
WHERE
   GREATEST(CEIL((L.Duration - DATEDIFF(CURDATE(), L.ApprovalDate)) / 30), 0)
<= 6;
```

Result Grid Filter Rows: Search Export:

CustomerID	FirstName	LastName	LoanID	RemainingDurationMont...
1	John	Doe	1	0
1	John	Doe	26	1
2	Jane	Smith	2	0
2	Jane	Smith	27	1
3	Michael	Johnson	3	0
3	Michael	Johnson	28	1
4	Emily	Williams	4	0
4	Emily	Williams	29	1
5	Christopher	Brown	5	0

Result 38